

Countdown Timer

Accurate timing from an AT90S1200 micro

Design by Andy Morell

andy@morell.freemove.co.uk

This project is simple to build and use yet accurate enough to provide a count down from any time interval anywhere from ninety-nine minutes down to one minute. The bright two digit display shows the current minutes left, and there is a piezo buzzer to indicate when the count reaches zero!

Main Features

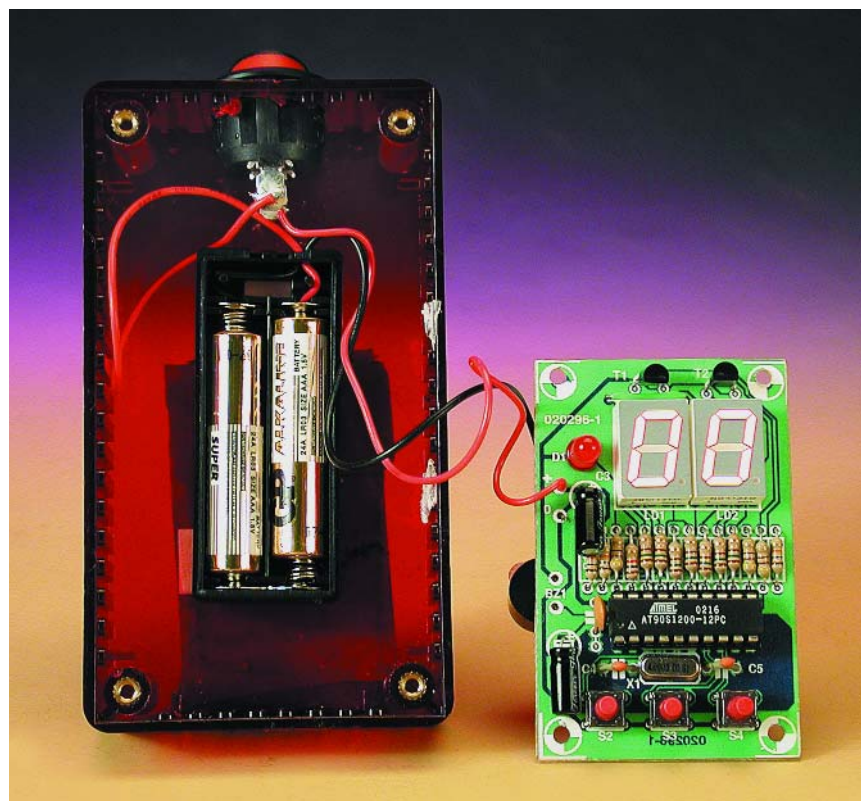
- Countdown period
1 minute to 99 minutes
- 2-digit 7-segment LED display
- 3-button control
- Low power consumption
- 3-volts battery supply
- AT90S1200 microcontroller chip

To use the timer is simplicity itself. First, select the time you require, by pressing the two appropriate push buttons. For example, press the unit button once for 1×1 minute, and the tens of unit button, say, three times for 3×10 minutes. There you are: a 31-minute countdown period. As you press the two switches, the number will show on the double-digit display. Both buttons will allow you to loop continuously from nought to nine, and back to nought.

Having entered the required time, you simply press the Start button. Immediately, a red LED will come on and the down count will commence. The displays start to flash when time-up is reached. A sleep function is automatically entered after 1 minute of buzzer activity. A new countdown cycle requires the microcontroller to be reset which means you have to switch the timer off and on again.

How the hardware works

The circuit diagram of the Countdown Timer is shown in **Figure 1**. With an AT90S1200



microcontroller running at 4 MHz and using one of its built in timers, (Timer 0), it is possible to get an accurate time period of one second, and so minutes, etc. The microcontroller clock frequency is generated using a dead standard 4-MHz quartz crystal and the usual pair of small

capacitors connected to the oscillator pins of the 90S1200.

The AT901200 microcontroller, is cheap, readily available, and easy to use. On top of that, it can work off a 3-volt battery supply, as we demonstrate here.

The reset pulse is generated by

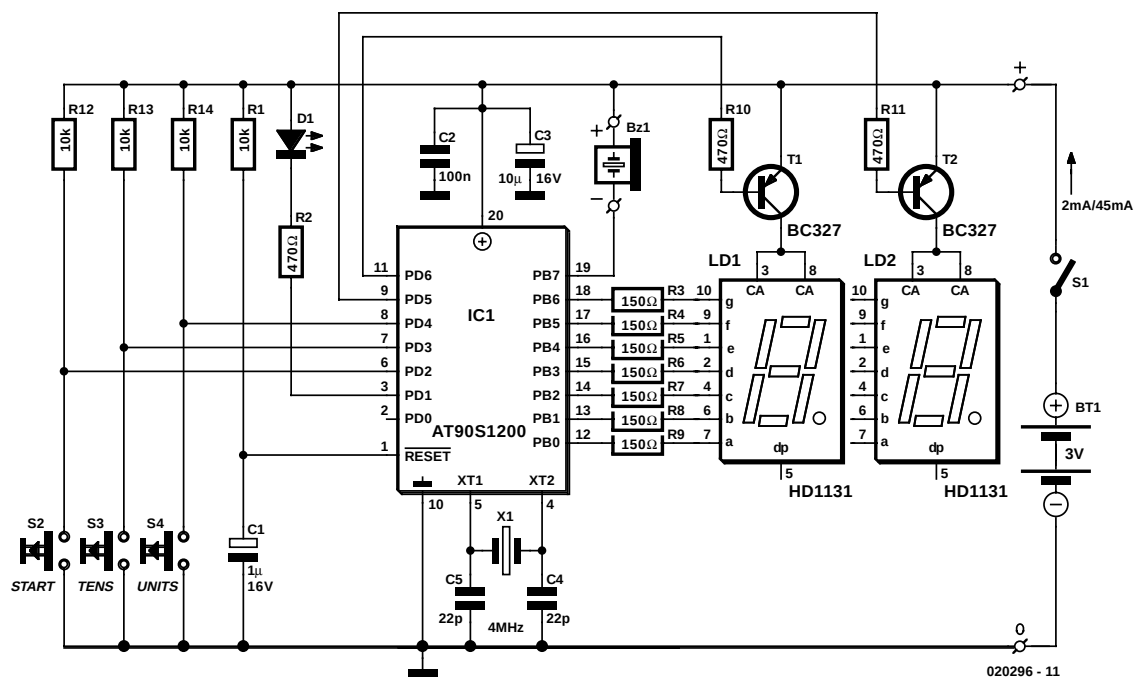


Figure 1. Circuit diagram of the Countdown Timer. A fair degree of component cutting has been achieved through the use of a microcontroller.

R1-C1, while PNP transistors T1 and T2 control the common anode terminals (CA) of the HD1131O displays. There are seven 100-Ω to 150-Ω resistors limiting the current through each of the seven segments of the active display. The actual value of the resistors will depend on the display brightness you require, set off against the current consumption (i.e., battery life), so a compromise will need to be found. Initially, we recommend using 150 Ω resistors.

There are also three 10-kΩ pull-up resistors for the three pushbutton switches. The logic High (+3 V) level created by these resistors helps prevent spurious triggering of the microcontroller input pins PD2, PD3 and PD4.

Finally, a small piezo buzzer of the 'active' (DC) variety is used. This device is switched on directly by the PB7 microcontroller pin and is sure to give an adequate sound level when it is required to.

How the software works

A flowchart of the main program running inside the AT90S1200 micro is shown in **Figure 2**. The structure of the interrupt routine is illustrated separately in **Figure 3**.

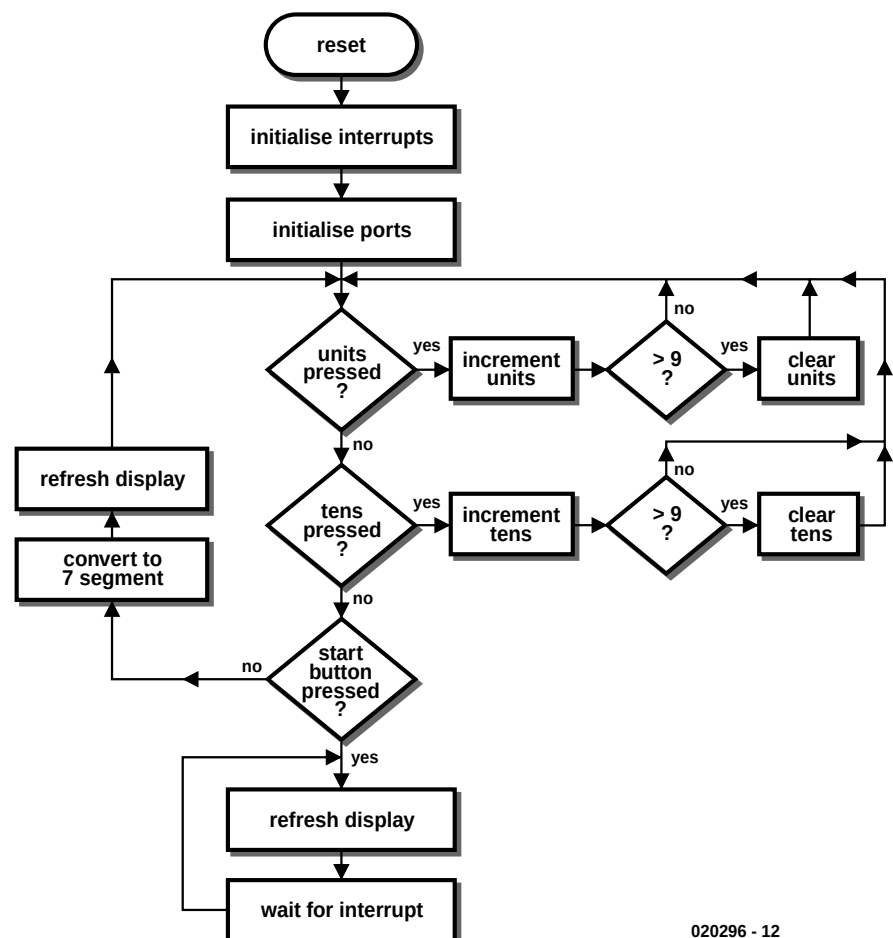


Figure 2. Main program flowchart.

To be able to understand the discussion below, it is useful to have the source code listing of the program available on paper or on your computer screen. The relevant file is available on floppy disk as order code **020296-11**, or as a Free Download from our website under the same number.

Upon Reset, the microcontroller sets up the stack pointer to enable it to store addresses etc., when calling subroutines, especially in the case of interrupts, which are used in this design.

As stated earlier, we use the built in timer Timer 0. This calls for the Timer 0 Control Register, TCCR0, to be set up. Division of the clock frequency by 256 allows us to generate an interrupt every 1 second. This is done by pre-loading the TCCR0 register with 0b00000100, or 04 in (hexa)decimal.

Setting up the Timer 0 interrupt to occur is the next task. This is done by setting bit 1 (TOIE0) in the TIMSK register. Now the Timer 0 interrupt is set up and will occur every 1 second using the 4-MHz crystal.

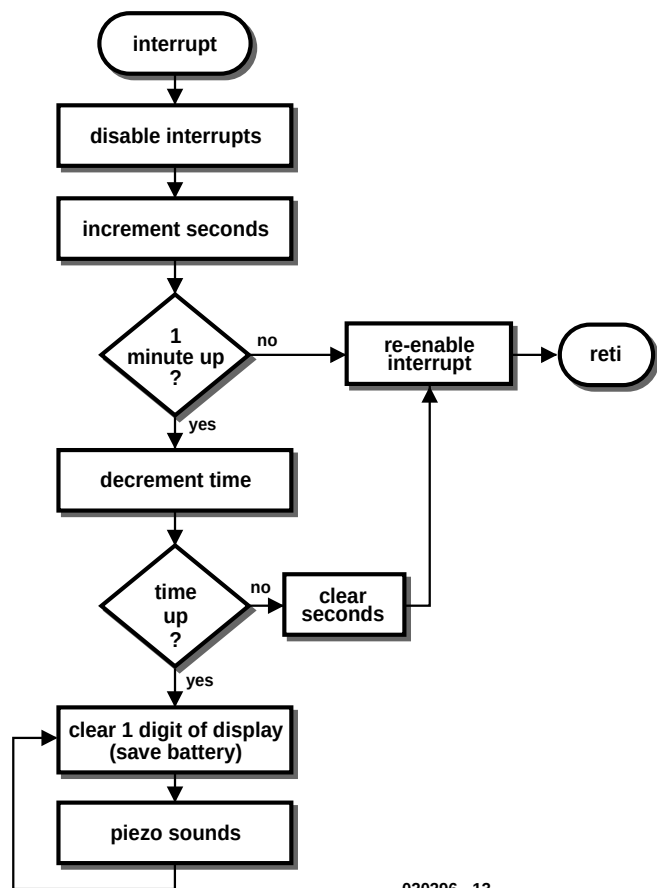
On interrupt the routine display DISPLAY_DIGITS is called. It is here that 61 interrupts are counted in a register called time_out.

Here is the relevant part of that routine:

DISPLAY_DIGITS:

```
IN      SAVE_STATUS, SREG
CPI     FLAG_1,$FF
BRNE ALARM_NOT_SET

INC     TIME_OUT
CPI     TIME_OUT, 61 ;1 SECOND
BREQ TIMER_OUT
```



020296 - 13

Figure 3. Interrupt routine flowchart.

We don't actually want the interrupt to occur just yet, and so we will now Inhibit it with the instruction 'CLI' (Clear Interrupt). Now the various ports are set up to their respective

input and output state.

You can easily see how the software for the pushbuttons works. If you press the 'units' button then the routine for incrementing the units is

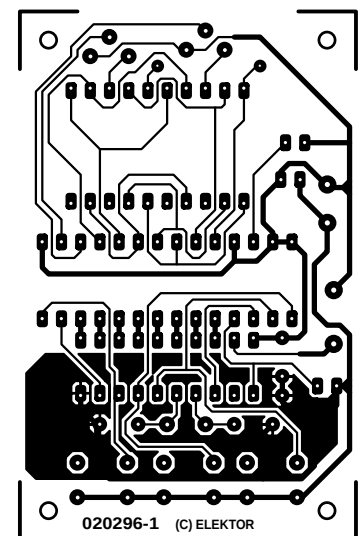
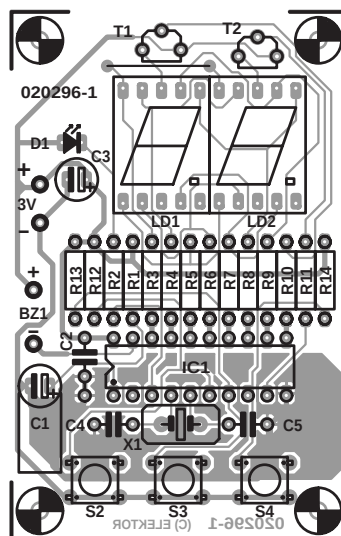
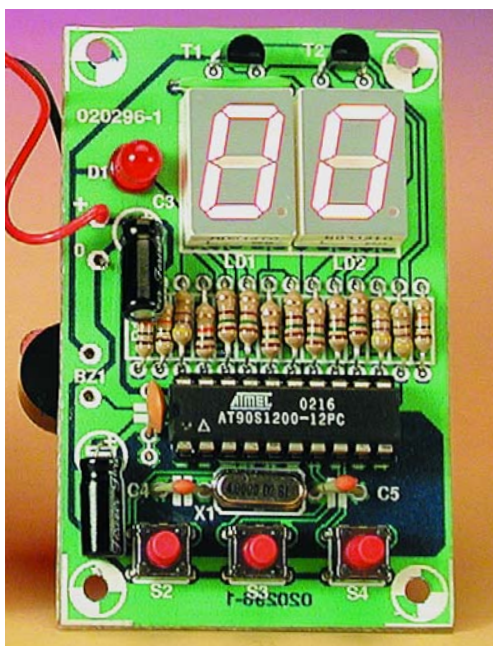


Figure 4. PCB design for the Countdown Timer (board available from The PCBShop).

COMPONENTS LIST

Resistors:

R1,R12,R13,R14 = 10k Ω
R2,R10,R11 = 470 Ω
R3-R9 = 100 Ω to 150 Ω (see text)

Capacitors:

C1 = 1 μ F 16V radial
C2 = 100nF
C3 = 10 μ F 16V radial
C4,C5 = 22pF

Semiconductors:

D1 = LED, red, 3mm dia.

IC1 = AT90S1200-12PC, order code
020296-41
LD1,LD2 = HD1131O (see text)
T1, T2 = BC327

Miscellaneous:

BT1 = two 1.5V AA or AAA batteries
with holder
Bz1 = 5V DC buzzer
S1 = on/off switch
S2,S3,S4 = pushbutton with 1 make
contact (see text)
X1 = 4MHz quartz crystal (low-
profile model)

called. The same applies to the 'tens' button. It is only when you press the 'Start' button that the program branches to the count down section (eventually).

Before pressing the 'Start' button, the program is in a loop, and refresh of the display is achieved within this loop. There is also a 'Get Seven Segment' routine, which puts the appropriate pattern on the displays.

The program is then waiting for you to take your finger off the 'Start' button, so it may proceed to the next bit.

At this point, the interrupt is enabled, and the Interrupt Service Routine is then made accessible every one second. Now the program is in a continual loop, with refresh display, and of course the conversion to 7-segment contained here.

Interrupts will start occurring now, and the Interrupt Service Routine is called (Figure 3).

Not only is this where the down count is done, but it also checks if the count has reached zero, or not (this is done by checking if COUNT1 and COUNT2 are both 0. If this is not

the case, then the microcontroller continues to display the current time and also calls a Convert To BCD Routine. This is to stop the display showing Alpha-numeric characters, i.e., characters other than 0-9.

Note that the interrupt is disabled at the start of the ISR and re-enabled at the close of it. Eventually the ISR is left, and the program returns to the main loop described earlier. This will continue until COUNT1 and COUNT2 are both at 00.

There's more to discover and enjoy in the source code listings than can be described here, and the free source code file is recommended not only to those of you who can program their own chips but also to AVR programmers looking for new ideas.

How to build the timer

The copper track layout and component mounting plan of the small single-sided PCB designed for the Countdown Timer are shown in **Figure 4**. This PCB is not available ready-made through our Readers Services but it may still be ordered

from The PCBShop operated by our business partner EuroCircuits.

Start the construction with fitting the single wire link above display LD1. Next, fit the resistors, and then the capacitors (watch the polarity of C1 and C3). After that, insert a 20-pin DIL IC socket, followed by the crystal. The piezo buzzer and the battery are external to the board and connected up by light-duty flexible wire.

The HD1131 common-anode display comes in several versions, the suffix indicating the colour. Here the 'O' super-red device is recommended. The luminosity may be indicated separately in the device type code, for example, by an 'L'. By choosing a really bright version, a good trade-off can be achieved between current consumption and readability. In this context, see also the notes on the values of R3-R9.

Depending on the enclosure you wish to use, the LED and the three pushbuttons may have to be mounted a little off the board surface to make sure they are flush with the enclosure top panel. In the case of the switches, this may require lengthening their pins with pieces of stiff wire.

Then solder in the two BC327 transistors and the displays. As with the LED and the switches, you may wish to raise the displays from the board surface by inserting them in SIL socket strips. There are some vero-pins for the battery, the buzzer and the on/off switch, which are panel mounted. Note that the batteries are 2 x 'AAA' cells, which are perfectly adequate for many hours of practical use.

(020296-1)

Free Downloads

– source code and HEX files. File number:
020296-11.zip

– PCB layout in PDF format. File number:
020296-1.zip

www.elektor-electronics.co.uk/dl/dl.htm,
select month of publication.